Understanding and Improving Laplacian Positional Encodings For Temporal GNNs

Yaniv Galron[1] (✉), Fabrizio Frasca[1], Haggai Maron[1,4], Eran Treister[2], and Moshe Eliasof[3]

[1] Technion – Israel Institute of Technology
`yaniv.galron@campus.technion.ac.il`
[2] Ben-Gurion University of the Negev
[3] University of Cambridge
[4] NVIDIA

Abstract. Temporal graph learning has applications in recommendation systems, traffic forecasting, and social network analysis. Although multiple architectures have been introduced, progress in positional encoding for temporal graphs remains limited. Extending static Laplacian eigenvector approaches to temporal graphs through the supra-Laplacian has shown promise, but also poses key challenges: high eigendecomposition costs, limited theoretical understanding, and ambiguity about when and how to apply these encodings. In this paper, we address these issues by (1) offering a theoretical framework that connects supra-Laplacian encodings to per-time-slice encodings, highlighting the benefits of leveraging additional temporal connectivity, (2) introducing novel methods to reduce the computational overhead, achieving up to 56x faster runtimes while scaling to graphs with 50,000 active nodes, and (3) conducting an extensive experimental study to identify which models, tasks, and datasets benefit most from these encodings. Our findings reveal that while positional encodings can significantly boost performance in certain scenarios, their effectiveness varies across different models. The supplementary materials and code are available at `https://github.com/YanivDorGalron/SLPE`.

Keywords: Temporal Graphs · Positional Encodings · Graph Laplacian

1 Introduction

Temporal Graph Neural Networks (TGNNs) have emerged as a state-of-the-art paradigm for learning on dynamic graphs [29,17,38,34,30,3,36,11]. By simultaneously capturing evolving temporal dynamics and underlying graph structure, TGNNs have achieved remarkable performance across applications like temporal link prediction [27,7,45], node classification [35], edge classification [24] or temporal clustering [22].

Positional Encoding (PE) techniques, fundamental to the success of Transformer architectures [37,10,4], enhance representational capacity by embedding crucial positional information within sequential and temporal data. In static

graph contexts, positional encodings — particularly the Laplacian Positional Encoding (LPE) [2,5] derived from the spectral decomposition of the graph Laplacian — have demonstrated significant benefits in injecting structural information and in consistently elevating overall performance across node classification, link prediction and other graph learning tasks [6,5,20].

Despite their potential benefits, positional encodings for temporal graphs remain largely underexplored. A recent advancement has been the adaptation of LPEs for TGNNs through the novel application of supra-Laplacian eigenvectors [14]. The supra-Laplacian [18,43,21,9,33] extends traditional graph Laplacian frameworks by incorporating temporal connectivity between time steps, thereby elegantly capturing both intra-layer structural and inter-layer temporal dynamics. This approach enriches positional encodings with temporal information and has empirically demonstrated improved downstream performance.

However, the adoption of supra-Laplacian based PEs (SLPEs) presents several substantial challenges that warrant a thorough study. First, the theoretical underpinnings and properties of these novel encodings—and their specific relevance to temporal graph learning—remain insufficiently characterized, hampering our understanding of their effectiveness. Second, computing the eigendecomposition of the supra-Laplacian introduces considerable computational overhead due to the increased dimensionality from temporal connections. Third, while initial research [14] demonstrated promising results with specific transformer-based TGNN architecture and datasets, the generalizability of SLPEs across diverse architectural frameworks and learning tasks remains an open question.

Main Contributions. This paper systematically addresses the three aforementioned gaps to advance both the theoretical and practical understanding of Laplacian-based PEs for TGNNs:

1. We develop a theoretical analysis of supra-Laplacian PEs (SLPEs) as compared to single-layer Laplacian PEs (LPEs), and discuss the increased expressive power given by the supra-graph representation.
2. We introduce a computationally efficient framework for calculating SLPEs through approximate eigenvector computation, shown in Figure 1.
3. We present extensive empirical evaluations across multiple Laplacian-based PEs, feature initializations, and architectural paradigms (message-passing- and transformer-based TGNNs), culminating in actionable practical guidelines.

2   Related Work

We now provide an overview of relevant topics to our work, namely TGNNs and the use of Laplacian Positional Encodings in graph learning.

Temporal Graph Neural Networks. TGNNs operate on both Continuous-Time Dynamic Graphs (CTDGs) [29,11,17,34] and Discrete-Time Dynamic Graphs (DTDGs) [32,41], with efforts to bridge the two domains [34,12]. For DTDGs,
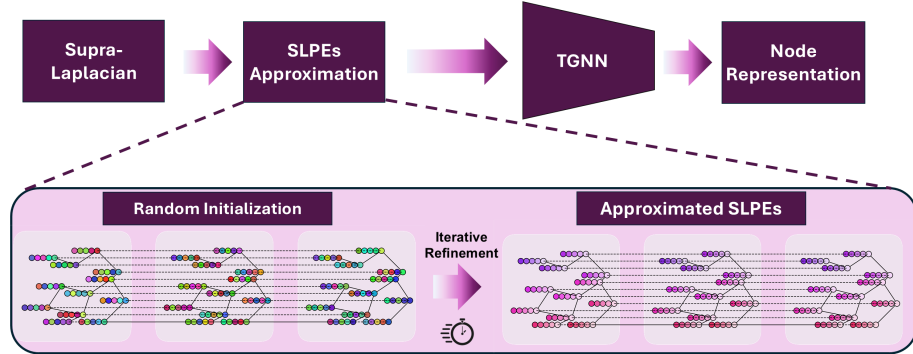
Fig. 1: An overview of our proposed fast SLPEs computation procedure. The SLPEs are generated via iterative solvers applied on the supra-Laplacian. Those start with random initialization and apply iterative refinement toward smooth PEs that act as the node representations.

early methods like EGCN [25] use a Recurrent Neural Networks approach to apply a Graph Convolutional Network (GCN) over time. HTGN [42] leverages hyperbolic geometry to model complex, hierarchical structures in evolving networks. For CTDGs, pioneering methods like DyRep [36] and JODIE [17] process timestamped edge streams, while TGAT [40] focuses on inductive representation learning. Temporal Graph Networks (TGNs) [29] generalize these approaches, encompassing DyRep, JODIE, and TGAT as special cases. In the context of PEs, [34] incorporated relative PEs into CTDGs by counting node appearances on temporal walks and [39] constructed PEs for CTDGs by leveraging the Poisson point process to efficiently estimate personalized interaction intensity.

Laplacian-Based Positional Encodings. Graph Laplacian eigenvectors [2] have gained widespread adoption as effective graph embedding tools. In static graph neural networks, these embeddings encode crucial structural information that demonstrably enhances GNN expressive power [5,28,23]. The recent work in [6] revealed that approximate eigenvectors—as well as their computation trajectories—can match or surpass the performance of exact eigenvectors. Meanwhile, [20] developed novel neural architectures invariant to inherent eigenvector symmetries, specifically sign flips and more general basis transformations. Important theoretical challenges were addressed by [13], which investigated the non-uniqueness and instability issues where minor perturbations to the Laplacian can produce substantially different eigenspaces. Building on these advances, [14] recently extended Laplacian PEs to the temporal graphs for TGNNs, by incorporating the supra-Laplacian into an innovative transformer-based architecture. A theoretical analysis of the supra-Laplacian for applications to graph learning is, however, still missing, as well as their practical effectiveness on other known message-passing-based TGNNs.

3   Supra-Laplacian PEs in Temporal Graphs

In this section, we first introduce the notation used in the paper, and then define the supra-Laplacian and SLPEs that were proposed in the recent work [14] to extend Graph Transformers to DTDGs. We consider SLPEs for MPNNs as well.

Notations and Definitions. We follow the setup of [15] where temporal graphs are represented as a sequence of snapshots $\mathbb{G} = \{G_1, ..., G_T\}$. Each snapshot $G_t = (\mathcal{V}_t, \mathcal{E}_t)$ contains nodes $\mathcal{V}_t$ and edges $\mathcal{E}_t$ at time step t. Nodes $v_t \in \mathcal{V}_t$ possess feature vectors $\mathbf{h}_{v_t} \in \mathbb{R}^d$, while edges $(u_t, v_t) \in \mathcal{E}_t$ may have associated features $\mathbf{w}_e \in \mathbb{R}^{d_e}$. Collectively, input node features are denoted as $\mathbf{H}_t \in \mathbb{R}^{|\mathcal{V}_t| \times d}$. In addition, we denote by $\mathbf{P}_t \in \mathbb{R}^{|\mathcal{V}_t| \times c}$ the PEs at time t. As is standard [5,6], the PEs are combined with input node features to form an initial representation $\tilde{\mathbf{H}}_t = [\mathbf{H}_t \| \mathbf{P}_t] \in \mathbb{R}^{|\mathcal{V}_t| \times (d+c)}$, where $\|$ denotes channel-wise concatenation.

Supra-Laplacian and -Adjacency. The supra-Laplacian [18,43,21,9,33] leverages the multi-layer structure of temporal graphs by constructing a block matrix representation of the graph sequence. For a temporal graph $\mathbb{G}$, the supra-Laplacian matrix $\mathbf{L}_{\mathrm{supra}} \in \mathbb{R}^{T|\mathcal{V}| \times T|\mathcal{V}|}$ is defined as:

$$\mathbf{L}_{\mathrm{supra}} = \mathbf{D}_{\mathrm{supra}} - \mathbf{A}_{\mathrm{supra}}, \tag{1}$$

where $\mathbf{A}_{\mathrm{supra}}$ is the supra-adjacency matrix, $\mathbf{D}_{\mathrm{supra}}$ is the corresponding degree matrix, and $|\mathcal{V}| = \max_{t=1,...,T} |\mathcal{V}_t|$. The supra-adjacency matrix is constructed by placing the adjacency matrices $\mathbf{A}_t$ of each snapshot G_t along the diagonal and adding inter-layer edges to model temporal dependencies, defined as:

$$\mathbf{A}_{\mathrm{supra}} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{B}_{12} & \cdots & \mathbf{B}_{1T} \\ \mathbf{B}_{21} & \mathbf{A}_2 & \cdots & \mathbf{B}_{2T} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{B}_{T1} & \mathbf{B}_{T2} & \cdots & \mathbf{A}_T \end{bmatrix}, \tag{2}$$

where $\mathbf{A}_t \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ is the adjacency matrix of snapshot G_t, and $\mathbf{B}_{ij} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ represents the inter-layer edges modeling temporal dependencies between snapshots G_i and G_j. Here, we set $\mathbf{B}_{ij}$ to be the identity matrix $\mathbf{I}$ when $|i - j| = 1$, i.e., the snapshots are connected to their immediate previous and next layers. This choice restricts interactions to the same node across adjacent time steps. We note that it limits the model's ability to capture complex temporal patterns– such as seasonal cycles or delayed cross-node effects (e.g., a traffic jam at one node influencing another later). Future work could explore richer $\mathbf{B}_{ij}$ matrices to better model such dynamics.

To work with evolving graphs, one can use a subset of the most recent snapshots of $\mathbb{G}$ where the window size represents the number of consecutive time steps or graph snapshots. For simplicity, we consider the window to be of size T.
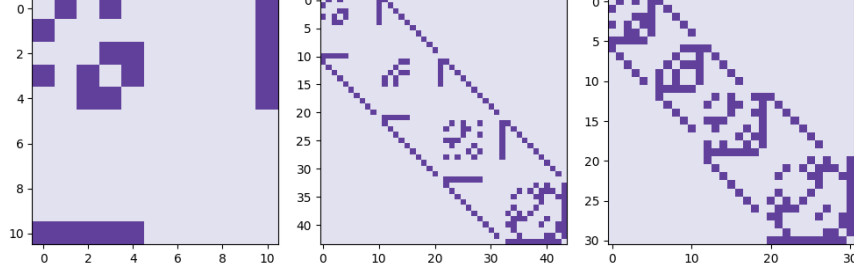
Fig. 2: Examples of adjacency matrices. Left: an adjacency matrix of a single snapshot with a global node connected only to the active nodes. Middle: the Supra-Adjacency matrix of shape $|\mathcal{V}|T \times |\mathcal{V}|T$ (middle). Right: the reduced Supra-Adjacency that contains only active nodes per snapshot.

Supra-Laplacian PEs (SLPEs). These PEs were firstly introduced in [14] where, for a node v at time t, they are derived from the eigenvectors of $\mathbf{L}_{\text{supra}}$ corresponding to the smallest eigenvalues. Let $\mathbf{X} \in \mathbb{R}^{T|\mathcal{V}| \times k}$ be the matrix of eigenvectors corresponding to the k smallest eigenvalues of $\mathbf{L}_{\text{supra}}$. The SLPEs $\mathbf{P}_t(v)$ for node v at time t is given by:

$$\mathbf{P}_t(v) = \mathbf{X}_{(t-1)|\mathcal{V}|+v,:k}, \tag{3}$$

where $\mathbf{X}_{(t-1)|\mathcal{V}|+v,:k}$ extracts the k-dimensional embedding corresponding to node v at time t. This encoding captures both the structural and temporal properties of the graph by leveraging the spectral properties of the supra-Laplacian. Furthermore, alongside these eigenvectors, the corresponding smallest eigenvalues of $\mathbf{L}_{\text{supra}}$ can also be concatenated to add further spectral information about the dynamic graph.

To enhance SLPEs, the work in [14] proposed two key modifications: (1) Global Node Integration: Each layer is augmented with a global node connected to all active nodes within that layer to better capture layer-wide activity and context (these nodes are considered part of $|\mathcal{V}|$); (2) Isolated Node Removal: Unconnected nodes are removed from the supra-adjacency matrix to eliminate the noise possibly introduced by considering uninformative eigenvectors.

4 Theoretical Understanding of Supra-Laplacian PEs

In this section, we enhance the theoretical understanding of SLPEs. First, we show how SLPEs inherently balance intra-layer structure preservation with inter-layer consistency through their smoothness. Second, we analyze the expressiveness advantages of the supra-adjacency matrix over layer-wise methods.
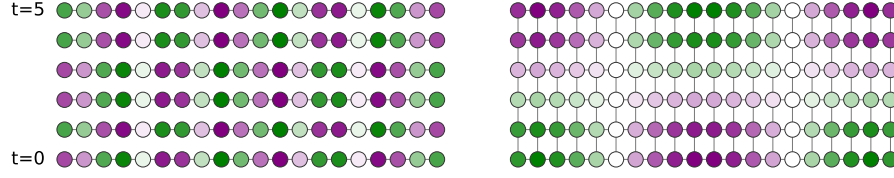
Fig. 3: Comparison of eigenvector smoothness in multilayer single path graphs. Left: Eigenvectors computed independently per layer, showing inconsistent sign assignments. Right: Eigenvector computed via the supra-Laplacian yield temporally smooth transitions. Graphs are single-path chains over time, highlighting the benefits of temporal regularization introduced by inter-layer edges.

4.1   Time and Space Smoothness with SLPEs

We now show that computing the d lowest supra-Laplacian eigenvectors is equivalent to minimizing an objective function that balances the preservation of layer-specific structure while promoting smooth transitions across layers via a penalty term. The proof of the proposition below appears in Appendix C.1.

Proposition 1. *(Supra-Laplacian PEs Smoothness). Let $\mathbb{G} = \{G_1, G_2, \ldots, G_T\}$ be a multilayer graph with T layers, where each layer G_t is represented by its adjacency matrix $\mathbf{A}_t$, the degree matrix $\mathbf{D}_t$, and the Laplacian matrix $\mathbf{L}_t$. In addition, $\mu > 0$ is a parameter that controls the weight of inter-layer connections. Then the eigenvectors of the supra-Laplacian matrix associated with $\mathbb{G}$ are the vectors $\mathbf{X}^{(t)} \in \mathbb{R}^{|\mathcal{V}| \times k}$ that minimize the following objective function:*

$$\min_{\mathbf{X}^{(t)}} \sum_{t=1}^{T} tr\left(\mathbf{X}^{(t)^T} \mathbf{L}_t \mathbf{X}^{(t)}\right) + \mu \sum_{t=2}^{T} \left\|\mathbf{X}^{(t)} - \mathbf{X}^{(t-1)}\right\|_F^2, \tag{4}$$

subject to $\mathbf{X}^T\mathbf{X} = \mathbf{I}$, where $\mathbf{X}$ is the concatenation of all matrices $\mathbf{X}^{(t)}$ and $\|\|_F$ is the Frobenius norm.

The minimization in Equation (4) shows that by using the supra-Laplacian eigenvectors, we achieve a balance between two key terms:

1. *Intra-layer smoothness*: The local connectivity structure of each layer G_t is preserved through the Laplacian quadratic form: $tr\left(\mathbf{X}^{(t)^T} \mathbf{L}_t \mathbf{X}^{(t)}\right)$. Minimizing it makes the eigenvectors smooth according to the Laplacian $\mathbf{L}_t$.
2. *Inter-layer consistency*: The penalty terms $\mu \left\|\mathbf{X}^{(t)} - \mathbf{X}^{(t-1)}\right\|_F^2$, in contrast, enforces smooth transitions between eigenvectors of adjacent layers.

The inter-layer consistency promoted by the smoothness term not only ensures the smoothness of inter-layer transitions but also encourages consistent sign assignments for eigenvectors, as can be seen in Figure 3. This is in contrast to independently computed eigenvectors for each layer, where consecutive eigendecompositions can lead to sign differences in each realization.

4.2   The Expressiveness Benefits of Using the Supra-Adjacency

Here, we shed light on the usefulness of considering the (multi-layer) supra-adjacency matrix rather than a single layer-wise approach. A key tool in our analysis is the Supra-Weisfeiler-Lehman (Supra-WL) test, which we define to extend the classical WL isomorphism test to snapshot-based temporal graphs represented as supra-graphs. Supra-WL operates by iteratively refining node colors in the supra-graph: (i) it starts by assigning a constant color $\bar{c}$ to each node in the supra-graph $\mathbb{G}(\tau)$, or one which uniquely encodes node features, if available; (ii) it refines these colors by injectively hashing the current color, the colors of its temporal neighbors, as well as the multiset of colors of its spatial neighbors within the same layer $G(t)$:

$$C_{v,t}^{(l+1)} = \text{HASH}\left(C_{v,t}^{(l)}, C_{v,t-1}^{(l)}, C_{v,t+1}^{(l)}, \{\!\!\{(C_{u,t}^{(l)}, e_{u,v,t}, t)|(u,v,t) \in G(t)\}\!\!\}\right) \quad (5)$$

where, for $t = 0$ and $t = \tau$ we have $C_{v,t-1}^{(l)} = C_{v,t+1}^{(l)} = \bar{\bar{c}}$. The test is applied in parallel to two temporal graphs; it terminates when the multisets of node colors for the two supra-graphs diverge, indicating non-isomorphism. If the colors stabilize without divergence, the test is inconclusive.

To understand the importance of the information contained in the supra-adjacency, we compare the Supra-WL to Layer-WL, a simple extension of WL to snapshot-based temporal graphs. Layer-WL runs 1-WL color refinement steps independently on each graph snapshot $G(t)$, comparing the overall multisets of node colors thereon. We now present a critical distinction between the two algorithms, shedding light on the enhanced capabilities of the Supra-WL test and, by extension, models considering supra-adjacency information.

Proposition 2. *(Supra-WL $\sqsubset$ Layer-WL). Supra-WL is strictly more powerful than Layer-WL in distinguishing non-isomorphic DTDGs.*

The proof of this proposition is deferred to Appendix C.2, and involves exhibiting a pair on non-isomorphic DTDGs that are distinguished by Supra-WL but not by Layer-WL, reported in Fig 4. This example emphasizes how treating layers as interconnected rather than independent is essential for capturing structural differences in temporal or multi-layer graphs that would otherwise go unnoticed.

5   Efficient Computation of Supra-Laplacian PEs

Focusing on enhancing the efficiency of eigendecompositions for the scalable computation of (S)LPEs, we propose and evaluate several strategies centered around iterative eigendecomposition and trajectory-based analysis. The overview of the proposed procedure is illustrated in Fig 1.

First, we propose exploring the use of the Lanczos method [19], an iterative algorithm designed for large, sparse, symmetric matrices like graph Laplacians. It constructs a Krylov subspace and diagonalizes a smaller tridiagonal matrix, achieving exact solutions with sufficient iterations. Second, questioning
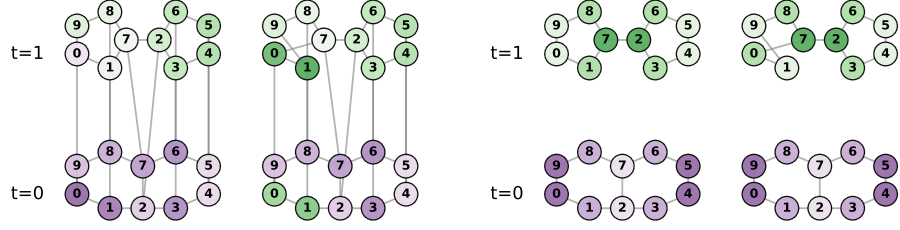
Fig. 4: Left: The Supra-WL test distinguishes two non-isomorphic temporal graphs by leveraging inter-layer edges, shown as vertical connections between time steps. Right: Layer-WL fails to distinguish the graphs as it considers only intra-layer structures.

the necessity of exact eigendecomposition, we propose to utilize solutions derived from the Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG) method [16]. LOBPCG is another iterative algorithm specifically engineered to efficiently compute a limited number of extreme eigenvalues and eigenvectors. LOBPCG offers memory efficiency, particularly when only a subset of eigenvectors is required.

Furthermore, to leverage the information computed during iterative eigendecomposition solvers, we introduce a trajectory-based approach. Inspired by the work of [6] on static graphs, we extend this technique to temporal graphs. This approach recognizes that intermediate results from iterative solvers can be valuable and proposes to concatenate these intermediate results rather than solely relying on the achieved approximated solution. To address the inherent sign ambiguity of eigenvectors, for each eigenvector, we randomly choose a sign ($+1$ or -1) as was done in [5] and consistently apply this sign across all iterations of its trajectory. Specifically, at each iteration k of the eigendecomposition algorithm, let $\mathbf{U}^{(k)} \in \mathbb{R}^{n \times k}$ be the matrix of eigenvectors and $\mathbf{\Lambda}^{(k)}$ be the corresponding eigenvalues. Our trajectory-based approach constructs concatenated representations as follows: $\mathbf{U}_{\text{traj}} = \left[\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(K)} \right], \mathbf{\Lambda}_{\text{traj}} = \left[\mathbf{\Lambda}^{(1)}, \mathbf{\Lambda}^{(2)}, \ldots, \mathbf{\Lambda}^{(K)} \right]$. Here, K represents the total number of iterations, determined by either convergence criteria or a predefined early stopping point. This concatenated form aims to capture the evolution of eigenvectors and eigenvalues across iterations, potentially providing a richer representation of the temporal graph dynamics.

6   Experiments

In this section, we present a detailed evaluation of Laplacian-based PE variants for temporal graphs with various architectures. Our experiments aim to assess the impact of these approaches on downstream performance across diverse real-world datasets and feature configurations. In particular, our focus is on comparing standard graph representations, such as single snapshot-based graphs,

with supra-graph representations, which combine multiple snapshots into a single supra-graph. Additionally, we investigate the effects of using iterative and approximate solvers for eigendecomposition in these models. We aim to address the following questions:

1. Do Laplacian-based PEs enhance the performance of TGNNs, in general?
2. Which Laplacian-based PE scheme is best suited for TGNNs?
3. How do node features impact the performance of the Laplacian-based PEs?
4. What are the computational benefits of *approximate* Laplacian-based PEs?

Full details on experiments and additional results are provided in Appendix D.

6.1   Experimental Setup

We evaluate four temporal graph models: EGCN [25], GRUGCN [31], HTGN [42], and SLATE [14]. Further details about these models are provided in Appendix B. In our implementation of SLATE, we separated the SLPE component from the architecture, which allowed us to investigate various alternative PEs. Additionally, we performed a detailed time analysis comparing the full eigendecomposition, the Lanczos method (which we ran until convergence), and the LOBPCG method for computing the first 8 eigenvectors on both synthetic and real-world datasets. An overview of the proposed procedure is illustrated in Figure 1.

Node features. We process the datasets under various feature configurations. We start with one-hot encodings, a widely used approach in the literature [25,15], that assigns a unique identifier to each node in the observed snapshot. However, we argue that this method may not be realistic for dynamic graphs where the number of nodes can grow unpredictably over time. Additionally, one-hot encodings may be suboptimal in the presence of nodes with few interactions, as the parameters associated with such nodes could remain undertrained and lead to poorer generalization performance. Accordingly, we explore two additional feature encodings and their interplay with PEs. First, we experiment with uninformative, constant (zero) features, simulating the absence of node-specific information. Second, we study the impact of random node features. Although lacking any temporal and structural inductive bias, random features can, in principle, allow the model to identify nodes throughout its computations [1].

Laplacian-based PE variants. As a baseline, we report results without PEs (No PEs). In other cases, we compare several variants involving different PEs: SLPE and LPE, and different types of approximation: Exact eigenvalue computation *(E)*, Inexact computation *(I)*, and using the computation trajectory *(T)*, as explained in Section 5. We use the widely adopted Lanczos method for (E), appropriately run until convergence. LOBPCG is employed for (I). As for (T), we employ the full intermediate trajectory generated by the latter. We note that the modification to the graph mentioned in Section 3 and illustrated in Figure 2 are also done to the standard Laplacian prior to PE computation.

Table 1: AUC performances of models across datasets and configurations for one-hot features. The top three models are highlighted by First, Second, **Third**.

Dataset	Variant	EGCN	GRUGCN	HTGN	SLATE
CanParl	No PEs	85.56±0.27	67.10±1.54	87.59±0.69	56.22±1.31
	SLPE-E	83.53±1.59	**72.92±1.90**	**89.47±0.29**	59.32±0.63
	LPE-E	85.41±1.44	74.92±0.74	89.62±0.16	59.99±1.07
	SLPE-I	83.45±1.59	**72.71±1.41**	89.26±0.55	57.42±0.97
	LPE-I	**84.18±1.06**	71.53±0.73	89.61±0.19	56.96±0.69
	SLPE-T	82.46±1.12	65.54±1.81	88.90±0.77	**58.71±1.44**
	LPE-T	81.72±0.89	67.11±1.79	88.98±0.31	55.04±1.28
as733	No PEs	92.47±0.04	94.96±0.35	98.75±0.03	99.85±0.01
	SLPE-E	**93.54±0.87**	95.46±1.59	98.28±0.33	99.81±0.02
	LPE-E	94.00±0.88	96.93±0.13	**98.10±0.19**	99.84±0.01
	SLPE-I	93.99±1.37	95.07±0.69	97.81±0.21	99.84±0.01
	LPE-I	93.52±0.65	96.13±1.11	97.61±0.34	99.80±0.01
	SLPE-T	93.19±0.89	**96.75±0.11**	91.62±0.75	99.81±0.02
	LPE-T	92.03±0.20	96.79±0.40	86.92±1.43	99.81±0.01
dblp	No PEs	83.88±0.53	84.60±0.92	89.26±0.17	89.43±0.42
	SLPE-E	87.10±0.23	86.93±0.96	88.67±0.55	89.68±0.56
	LPE-E	82.57±0.60	86.89±0.70	**88.74±0.15**	89.25±0.28
	SLPE-I	86.66±0.34	**86.93±0.46**	88.77±0.12	89.38±0.42
	LPE-I	83.90±0.48	87.04±0.98	88.52±0.36	89.40±0.19
	SLPE-T	**85.57±0.38**	86.29±0.78	88.59±0.31	89.51±0.30
	LPE-T	80.67±0.60	86.70±0.62	88.08±0.37	**89.46±0.34**
enron10	No PEs	90.12±0.69	92.47±0.36	94.17±0.17	95.66±0.45
	SLPE-E	91.54±0.69	93.51±0.27	94.49±0.08	95.60±0.27
	LPE-E	**90.48±0.64**	93.34±0.83	**94.37±0.24**	95.49±0.33
	SLPE-I	91.40±0.75	93.63±0.13	94.45±0.54	95.49±0.31
	LPE-I	89.89±0.31	**93.45±0.48**	**94.37±0.20**	95.49±0.16
	SLPE-T	89.89±1.27	92.62±1.01	92.99±0.52	95.41±0.30
	LPE-T	88.13±0.97	92.90±0.59	93.36±0.30	95.43±0.16

Datasets, task, and performance metric. The mentioned models are tested on the real-world datasets: CanParl, as733, dblp, and enron10 [44,42]. Each represents dynamic graphs derived from snapshot-based observations (see Appendix A for more details). The task is (dynamic) link prediction in all cases. Performance is measured using the Area Under the Curve (AUC) metric, reported as the mean and standard deviation over five runs. The top three performing configu-
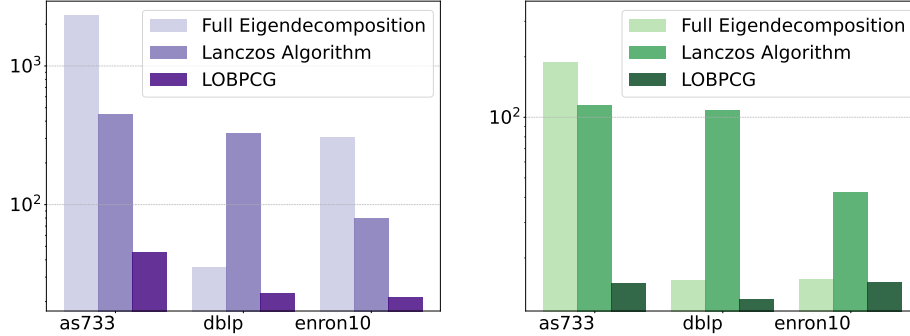
Fig. 5: Time (ms) performance comparison of Full Eigendecomposition, Lanczos, and LOBPCG methods on a real-world dataset presented by a Supra-Graph (left) and a single-layer-graph (right).

rations for each model-dataset pair are highlighted as First, Second, and **Third**, respectively, based on the mean test AUC score.

6.2   Results and Discussion

In this section we will presents the performance evaluation of various TGNN architectures across different datasets and PE schemes, focusing on the impact of Laplacian-based PEs and node feature configurations. We compare the effectiveness of one-hot, constant-zero, and random node features, with detailed results reported in Table 1, and additional results in Table 3 and Table 4 in Appendix D.2. Additionally, summarized performance metrics and comparisons across architectures and PE variants are provided in Table 2 and Table 3. We now address key questions regarding the efficacy of Laplacian-based PEs, the best-suited PE variants, the role of node features, and the computational efficiency.

Laplacian-based PEs & TGNNs (Q1). From Table 1, Table 3 and Table 4 we observe that in $\approx 70.8\%$ of the overall number of cases, using Laplacian-based PEs led to the top-scoring results (First), and that in $\approx 64.5\%$ of the experiments, all the top-three ranking models employ Laplacian-based PEs. Quantitatively, Table 5 in Appendix D.2 reports statistics on the absolute performance improvements induced by Laplacian-based PEs for each architecture. In all cases, except for HTGN, we observe positive median performance improvements, with the largest impact attained on SLATE (16.63%) and EGCN (8.20%). Importantly, we also observe how PEs are most useful when employing constant-zero node features, where they scored First in $^{13}/_{16}$ cases, as shown in Table 3. As for the other feature configurations, they ranked First in $^{11}/_{16}$ cases (one-hot) and $^{8}/_{16}$

Table 2: Average AUC (%) performance per feature and model when *Laplacian-based PEs are used* along with the difference between the max and min performance of different features for each model (Δ).

Feature	EGCN	GRUGCN	HTGN	SLATE	Avg
one-hot	87.87	86.75	91.73	85.66	88.00
random	81.72	80.89	88.59	77.63	82.21
constant	87.16	84.74	89.48	80.55	85.48
Δ	6.15	5.86	3.14	8.03	5.79

Table 3: Average AUC (%) performances across Laplacian-based PEs and feature inits.

Variant	one-hot	random	constant
No PEs	87.63	78.00	68.75
LPE-E	88.75	83.46	86.43
LPE-I	88.21	80.96	85.0
LPE-T	86.45	78.79	84.16
SLPE-E	88.74	83.57	86.83
SLPE-I	88.52	85.14	87.23
SLPE-T	87.37	81.33	83.24

cases (random). From Table 6 in Appendix D.2, we note, more specifically, positive median AUC improvements are more pronounced in the case of constant features, where even HTGN seems to generally benefit from PEs. Quantitative improvements are also recorded, on average, for EGCN and GRUGCN across all feature variants, as well as SLATE when not using one-hot features. We conclude that Laplacian-based PEs are generally useful in improving generalization performance, and they are more consistent when the model is not provided with node-identifying information.

Best Suited Laplacian-based PE variant for TGNNs (Q2). First, we compare SLPE variants to LPE variants. We found SLPE to perform better in 64.6% of considered cases, with an aggregated average improvement of 1.09% across models and datasets, as shown in Table 4. The improvement is more consistent, in particular in EGCN and SLATE, where the average absolute improvements are, resp., 3.91% and 0.66%. Notably, GRUGCN shows a small preference toward LPEs with an average performance difference of 0.30%. Next, we compare Exact variants with their Inexact counterparts. Overall, the former ones outperform the latter in 62.5% of cases, but the improvement is less pronounced in this case. As can be observed in Table 4, the distribution of performance differences is more dispersed, with median values close to zero in the case of HTGN and SLATE and only slightly in favor of Exact variants for EGCN and GRUGCN. These findings indicate that the faster Inexact variants have strong potential to be used in the development of more efficient processing pipelines. Finally, we compare variants overall, commenting on aggregated per-variant average performances as reported in Table 5. Across settings, SLPE-I leads with an average AUC of 86.96%, followed by SLPE-E with 86.38%, further confirming the suitability of approximate, iterative eigensolvers. Additional results separating different models, variants. and features can be seen in Table 6

Impact of features when coupled with Laplacian-based PEs (Q3). Refer to Table 2 for detailed per-model averages. When PEs are used, one-hot features outperform others with an aggregated average AUC of 88.00%. They excel across models

Table 4: AUC (%) Average and median performance differences between E and I variants per model and between SLPE and LPE variants.

Model	E - I		SLPE - LPE	
Diffs	Avg	Med [Q1, Q3]	Avg	Med [Q1, Q3]
EGCN	0.36	0.17 [-0.68, 0.57]	3.91	2.27 [0.70, 4.86]
GRUGCN	1.25	0.15 [-0.11, 1.43]	-0.30	-0.01 [-1.16, 0.69]
HTGN	0.003	0.02 [-0.08, 0.43]	0.08	0.04 [-0.21, 0.23]
SLATE	0.21	0.08 [-0.32, 0.36]	0.66	0.46 [-0.02, 3.77]
Mean Diff	0.46	0.04	1.09	0.69

Table 5: Average AUC (%) per variant across all models, datasets and features.

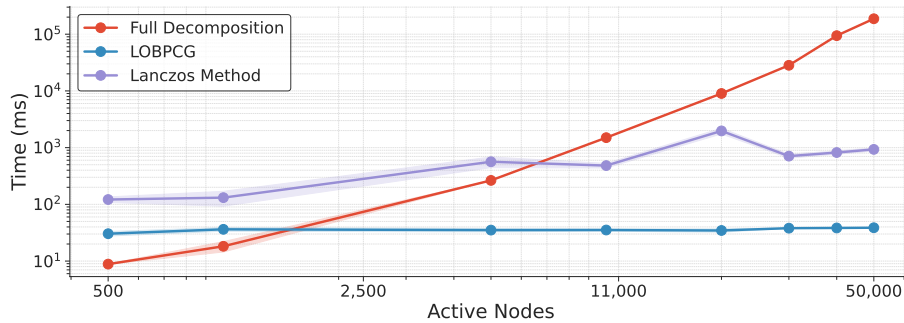| Variant	Average AUC
SLPE-I	86.96
SLPE-E	86.38
LPE-E	86.21
LPE-I	84.72
SLPE-T	83.98
LPE-T	83.13
No PEs	78.13

Fig. 6: Comparison of runtime (ms, log scale) for computing the first 8 eigenvectors of Barabási–Albert graphs using Full Eigendecomposition, Lanczos, and LOBPCG. LOBPCG and Lanczos scale substantially better than Full Decomposition, with LOBPCG exhibiting the most consistent performance across increasing numbers of active nodes.

(e.g., 91.73% for HTGN) and rank as the top performers in 18–23 cases per model. We argue that while one-hot encodings yield higher-performing models, those may be less practical for real-world scenarios where the number of nodes is unknown. Other features trail behind in both performance and frequency, with an average AUC of 85.48% for constant (zero) features and 82.21% for random features. A complementary angle to this discussion is offered by Table 3, where we report the average performance across models and datasets for each PE variant and feature choice. In agreement with our discussion in regards to Q1, we observe that PEs are most beneficial when using constant features. In addition to this, we note how the choice of PE variant is less impactful for one-hot features, while it leads to more result variability in the case of constant and random features. This effect is particularly pronounced in the latter case. In both settings, SLPEs achieve the best performance on average.

Computational Benefits of approximate Laplacian-based PEs (Q4). Our time measurements, depicted in Figure 5, show that LOBPCG is the fastest method for calculating the approximate Laplacian, consistently outperforming both Lanczos and Full Eigendecomposition in in experiments on regular and Supra-graphs. In an effort to extend our time comparisons to even larger graphs, we synthesized random Barabási–Albert graphs with up to 50,000 active nodes, and timed the different methods thereon — see Figure 6 for results. We observe that LOBPCG achieves a maximum speed-up of 56 times over Lanczos, with its efficiency advantage growing significantly as the effective size of the graph increases.

Table 6: Average AUC (%) performance across datasets. The top two scores for each model and feature combination are highlighted by First and Second.

Model	Variant	one-hot	randn	zeros	Model	Variant	one-hot	randn	zeros
EGCN	No PEs	88.01	81.43	50.0	HTGN	No PEs	92.44	**91.08**	85.37
	LPE-E	88.12	80.59	85.36		LPE-E	**92.71**	89.39	89.55
	LPE-I	87.87	73.74	85.2		LPE-I	92.53	89.2	88.99
	LPE-T	85.64	80.05	86.1		LPE-T	89.34	87.52	**89.82**
	SLPE-E	**88.93**	83.27	**88.79**		SLPE-E	**92.73**	88.94	89.28
	SLPE-I	**88.88**	**87.62**	**89.55**		SLPE-I	92.57	**89.52**	**89.76**
	SLPE-T	87.78	**85.08**	87.92		SLPE-T	90.52	86.99	89.44
GRUGCN	No PEs	84.78	79.68	84.14	SLATE	No PEs	85.29	59.81	55.49
	LPE-E	**88.02**	**83.68**	**88.96**		LPE-E	**86.14**	80.2	81.83
	LPE-I	87.04	80.4	84.56		LPE-I	85.41	80.49	81.23
	LPE-T	85.88	79.14	80.83		LPE-T	84.94	68.44	79.87
	SLPE-E	**87.2**	80.64	85.4		SLPE-E	**86.1**	**81.44**	**83.86**
	SLPE-I	87.08	**81.17**	**86.16**		SLPE-I	85.53	**82.25**	**83.43**
	SLPE-T	85.3	80.31	82.54		SLPE-T	85.86	72.96	73.05

Results Summary. In summary, our experiments demonstrate that integrating both LPEs and SLPEs into TGNNs generally enhances performance, especially when node features provide less discriminative information. When the node set is known in advance, one-hot features lead to the best performance, while constant features remain a solid alternative when coupled with Laplacian-based PEs, especially SLPEs. Specifically, regarding the choice of PEs: the SLPEs variants generally outperform LPEs except in the case of HTGN. Most notably, the SLPE-I variant emerges as a robust default, balancing high accuracy (86.96% average) with computational efficiency. Finally, our evaluation of eigenvector solvers reveals that approximate methods such as LOBPCG offer significant speed-ups, making them more suitable for large-scale graphs.

7 Conclusions

In this paper, we have thoroughly reviewed Laplacian-based PEs within the framework of TGNNs, providing a comprehensive understanding of their role and limitations. Our theoretical analysis of SLPEs reveals significant insights into their expressive power and connection to single-layer Laplacian PEs. In addition, we have demonstrated the practical implications of various PEs, providing actionable guidelines for practitioners seeking to optimize TGNN performance. Our findings highlight the benefits of incorporating Laplacian-based PEs and their interplay with node feature initialization schemes, underscoring how faster, approximate eigendecompositions can maintain a compelling tradeoff between run-time and model performance. We believe this work opens up interesting future research directions. These include exploring the study of further, more efficient eigensolvers for large graphs and the use of SLPEs in CTDGs.

Disclosure of Interests. The author declares no competing interests relevant to the content of this article.

References

1. Abboud, R., Ceylan, İ.İ., Grohe, M., Lukasiewicz, T.: The surprising power of graph neural networks with random node initialization. In: Proceedings of the Thirtieth International Joint Conference on Artifical Intelligence (IJCAI) (2021)
2. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput. **15**(6), 1373–1396 (Jun 2003)
3. Cong, W., Zhang, S., Kang, J., Yuan, B., Wu, H., Zhou, X., Tong, H., Mahdavi, M.: Do we really need complicated model architectures for temporal networks? In: The Eleventh International Conference on Learning Representations (2023), `https://openreview.net/forum?id=ayPPc0SyLv1`
4. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2021), `https://openreview.net/forum?id=YicbFdNTTy`
5. Dwivedi, V.P., Joshi, C.K., Luu, A.T., Laurent, T., Bengio, Y., Bresson, X.: Benchmarking graph neural networks. Journal of Machine Learning Research **24**(43), 1–48 (2023)

6. Eliasof, M., Frasca, F., Bevilacqua, B., Treister, E., Chechik, G., Maron, H.: Graph positional encoding via random feature propagation. In: International Conference on Machine Learning. pp. 9202–9223. PMLR (2023)

7. Fard, S.H., Ghassemi, M.: Temporal Link Prediction Using Graph Embedding Dynamics . In: 2023 IEEE Ninth Multimedia Big Data (BigMM). pp. 48–55. IEEE Computer Society, Los Alamitos, CA, USA (Dec 2023). `https://doi.org/10.1109/BigMM59094.2023.00014`, `https://doi.ieeecomputersociety.org/10.1109/BigMM59094.2023.00014`

8. Fey, M., Lenssen, J.E.: Fast graph representation learning with pytorch geometric (2019), `https://arxiv.org/abs/1903.02428`

9. Gómez, S., Díaz-Guilera, A., Gómez-Gardeñes, J., Pérez-Vicente, C.J., Moreno, Y., Arenas, A.: Diffusion dynamics on multiplex networks. Phys. Rev. Lett. **110**(2), 028701 (Jan 2013)

10. Heo, B., Park, S., Han, D., Yun, S.: Rotary position embedding for vision transformer. In: European Conference on Computer Vision. pp. 289–305. Springer (2024)

11. Huang, S., Poursafaei, F., Danovitch, J., Fey, M., Hu, W., Rossi, E., Leskovec, J., Bronstein, M., Rabusseau, G., Rabbany, R.: Temporal graph benchmark for machine learning on temporal graphs. Advances in Neural Information Processing Systems **36**, 2056–2073 (2023)

12. Huang, S., Poursafaei, F., Rabbany, R., Rabusseau, G., Rossi, E.: UTG: Towards a unified view of snapshot and event based models for temporal graphs. In: The Third Learning on Graphs Conference (2024), `https://openreview.net/forum?id=ZKHV6Cpsxg`

13. Huang, Y., Lu, W., Robinson, J., Yang, Y., Zhang, M., Jegelka, S., Li, P.: On the stability of expressive positional encodings for graphs. In: The Twelfth International Conference on Learning Representations (2024), `https://openreview.net/forum?id=xAqcJ9XoTf`

14. Karmim, Y., Lafon, M., Fournier-S'niehotta, R., THOME, N.: Supra-laplacian encoding for transformer on dynamic graphs. In: The Thirty-eighth Annual Conference on Neural Information Processing Systems (2024), `https://openreview.net/forum?id=vP9qAzr2Gw`

15. Kazemi, S.M., Goel, R., Jain, K., Kobyzev, I., Sethi, A., Forsyth, P., Poupart, P.: Representation learning for dynamic graphs: a survey. J. Mach. Learn. Res. **21**(1) (Jan 2020)

16. Knyazev, A.: Recent implementations, applications, and extensions of the locally optimal block preconditioned conjugate gradient method (lobpcg) (2017), `https://arxiv.org/abs/1708.08354`

17. Kumar, S., Zhang, X., Leskovec, J.: Predicting dynamic embedding trajectory in temporal interaction networks. KDD **2019**, 1269–1278 (Aug 2019)

18. Kuncheva, Z., Kounchev, O.: Spectral properties of the laplacian of temporal networks following a constant block jacobi model. Phys. Rev. E **109**, 064309 (Jun 2024). `https://doi.org/10.1103/PhysRevE.109.064309`, `https://link.aps.org/doi/10.1103/PhysRevE.109.064309`

19. Lanczos, C.: An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. J. Res. Natl. Bur. Stand. B **45**, 255–282 (1950). `https://doi.org/10.6028/jres.045.026`

20. Lim, D., Robinson, J.D., Zhao, L., Smidt, T., Sra, S., Maron, H., Jegelka, S.: Sign and basis invariant networks for spectral graph representation learning. In: The Eleventh International Conference on Learning Representations (2023), `https://openreview.net/forum?id=Q-UHqMorzil`

21. Lin, W., Zhou, S., Li, M., Chen, G.: Dismantling interdependent networks based on supra-laplacian energy. In: Science of Cyber Security, pp. 205–213. Lecture notes in computer science, Springer International Publishing, Cham (2021)

22. Liu, M., Liu, Y., Liang, K., Tu, W., Wang, S., Zhou, S., Liu, X.: Deep temporal graph clustering. arXiv [cs.LG] (May 2023)

23. Maskey, S., Parviz, A., Thiessen, M., Stärk, H., Sadikaj, Y., Maron, H.: Generalized laplacian positional encoding for graph representation learning. In: NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations (2022), `https://openreview.net/forum?id=BNhhZwAlVNC`

24. Ozmen, M., Markovich, T.: Recent link classification on temporal graphs using graph profiler. Transactions on Machine Learning Research (2024), `https://openreview.net/forum?id=BTgHh0gSSc`

25. Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T., Leiserson, C.: Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 5363–5370 (2020)

26. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library (2019), `https://arxiv.org/abs/1912.01703`

27. Qin, M., Yeung, D.Y.: Temporal link prediction: A unified framework, taxonomy, and review. ACM Comput. Surv. **56**(4) (Nov 2023). `https://doi.org/10.1145/3625820`, `https://doi.org/10.1145/3625820`

28. Rampášek, L., Galkin, M., Dwivedi, V.P., Luu, A.T., Wolf, G., Beaini, D.: Recipe for a General, Powerful, Scalable Graph Transformer. Advances in Neural Information Processing Systems **35** (2022)

29. Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., Bronstein, M.: Temporal graph networks for deep learning on dynamic graphs. In: ICML 2020 Workshop on Graph Representation Learning (2020)

30. Sato, K., Oka, M., Barrat, A., Cattuto, C.: DyANE: Dynamics-aware node embedding for temporal networks. arXiv [physics.soc-ph] (Sep 2019)

31. Seo, Y., Defferrard, M., Vandergheynst, P., Bresson, X.: Structured sequence modeling with graph convolutional recurrent networks (2017), `https://openreview.net/forum?id=S19eAF9ee`

32. Skarding, J., Gabrys, B., Musial, K.: Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. IEEE Access **9**, 79143–79168 (2021). `https://doi.org/10.1109/ACCESS.2021.3082932`

33. Solé-Ribalta, A., De Domenico, M., Kouvaris, N.E., Díaz-Guilera, A., Gómez, S., Arenas, A.: Spectral properties of the laplacian of multiplex networks. Phys. Rev. E **88**, 032807 (Sep 2013). `https://doi.org/10.1103/PhysRevE.88.032807`, `https://link.aps.org/doi/10.1103/PhysRevE.88.032807`

34. Souza, A.H., Mesquita, D., Kaski, S., Garg, V.K.: Provably expressive temporal graph networks. In: Oh, A.H., Agarwal, A., Belgrave, D., Cho, K. (eds.) Advances in Neural Information Processing Systems (2022), `https://openreview.net/forum?id=MwSXgQSxL5s`

35. Sun, J., Gu, M., Yeh, C.C.M., Fan, Y., Chowdhary, G., Zhang, W.: Dynamic graph node classification via time augmentation. In: 2022 IEEE International Conference on Big Data (Big Data). pp. 800–805. IEEE (2022)

36. Trivedi, R., Farajtabar, M., Biswal, P., Zha, H.: Dyrep: Learning representations over dynamic graphs. In: International Conference on Learning Representations (2019), `https://openreview.net/forum?id=HyePrhR5KX`
37. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
38. Wang, Y., Chang, Y.Y., Liu, Y., Leskovec, J., Li, P.: Inductive representation learning in temporal networks via causal anonymous walks. In: International Conference on Learning Representations (2021), `https://openreview.net/forum?id=KYPz4YsCPj`
39. Wang, Z., Zhou, S., Chen, J., Zhang, Z., Hu, B., Feng, Y., Chen, C., Wang, C.: Dynamic graph transformer with correlated spatial-temporal positional encoding. In: Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining (2025)
40. da Xu, chuanwei ruan, evren korpeoglu, sushant kumar, kannan achan: Inductive representation learning on temporal graphs. In: International Conference on Learning Representations (ICLR) (2020)
41. Yang, L., Chatelain, C., Adam, S.: Dynamic graph representation learning with neural networks: A survey. IEEE Access **12**, 43460–43484 (2024)
42. Yang, M., Zhou, M., Kalander, M., Huang, Z., King, I.: Discrete-time temporal network embedding via implicit hierarchical learning in hyperbolic space. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. pp. 1975–1985 (2021)
43. Yang, Y., Tu, L., Guo, T., Chen, J.: Spectral properties of supra-laplacian for partially interdependent networks. Appl. Math. Comput. **365**(124740), 124740 (Jan 2020)
44. Yu, L., Sun, L., Du, B., Lv, W.: Towards better dynamic graph learning: New architecture and unified library. Advances in Neural Information Processing Systems **36**, 67686–67700 (2023)
45. Zhang, X., Wang, Y., Wang, X., Zhang, M.: Efficient neural common neighbor for temporal graph link prediction (2024), `https://arxiv.org/abs/2406.07926`